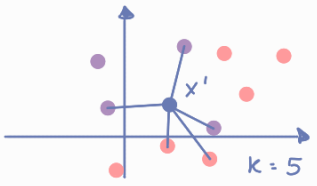
K-Nearest Neighbors (KNN)

Consider the usual classification problem $f: X \longrightarrow C$
with a dataset $D = \{(x_n, t_n)_{n=1}^N\}$

Classification with K-NN:
1. find the k nearest neighbors of the new instance x'
2. Assign to x the most common label among the majority of neighbors



$x' \notin D$

Likelihood of class c for new instance x':

$$P(c \mid x', D, K) = \frac{1}{K} \sum_{x_n \in N_K(x_n, D)} I(t_n = c)$$

input · dataset · number of neighbors · neighbors

in order to predict the class we will use the majority voting
● will be classified as ●

$P(● \mid ●, D, K=5) = 3/5$
$P(● \mid ●, D, K=5) = 1 - 3/5 = 2/5$

with $N_k(x_n, D)$ the k nearest points to x_n and $I(\ell) = \begin{cases} 1 & \text{iff } \ell \text{ is true} \\ 0 & \text{iff } \ell \text{ is false} \end{cases}$

When K = 1 this method corresponds to a Voronoi tesselation.

Increasing k brings smoother regions (reducing overfitting)

Pros
1. No training phase

Cons
1. Requires storing all the dataset
2. Depends on the distance function (in many cases it is not easy to define a distance function)

Kernelized KNN

In KNN everything depends on the choice of the distance function.
The distance function can be written as a kernel

$$\| x - x_n \|^2 = x^T x + x_n^T x_n + 2 x^T x_n$$

can be kernelized using a kernel $k(x, x_n)$

> When experimenting with a complex method, you should compare it with KNN (baseline) because of its simplicit